COURSE SYLLABUS AND REPORT

Advanced Database Systems

Feliz Alberto Ribeiro Gouveia

Universidade Fernando Pessoa

Porto, 2012

Contents

1 Introduction

This report was prepared by the author to partially fulfill the requirements of the submission to the "agregação" jury at University Fernando Pessoa. The "agregado" title is awarded by University Fernando Pessoa as described in **Regulamento n.º 307/2008**, published in the Official Journal, 2nd series, N. 110 of 9 June 2008. As stated in the aforementioned regulation, the title is awarded to candidates in recognition of:

a) The quality of their academic, professional, scientific and pedagogical curriculum.

b) Their ability to do research.

c) Their ability to supervise and to carry on independently research work.

Part of the requirements consists in a detailed curriculum vita and a report about an academic course, a group of courses or a study plan. The candidate chose to present the later report about an academic master level course, the *Advanced Database Systems* course.

This report describes the *Advanced Database Systems* course from the *Engenharia Informática* (Computer Science) degree at University Fernando Pessoa, Porto. It identifies in particular the course's goals and the skills it provides to students, its outcomes, the syllabus and the lesson plans. It further presents and discusses a lecture plan, exercises, and assessment methods. Finally it compares the course with similar offerings at other institutions, national and international.

The report has the following sections:

Section 1 this section.

Section 2 introduces the course's context, describing briefly the Portuguese dual degree higher education system, and the Engenharia Informática degree at University Fernando Pessoa.

Section 3 presents and describes the course, its goals, its outcomes and skills it provides to students. It describes in detail the course's syllabus, and the lecture plan with bibliography, assessment activities and resources.

Section 4 presents the delivery and lecturing methodology.

Section 5 describes the assessment goals and methods.

Section 6 lists and analyzes similar course offerings in other institutions.

Section 7 lists research groups and research resources available

Section 8 lists the references and the bibliography.

Section 9 presents examples of assessment questions.

Where appropriate, references to local or national regulations, methods or requirements are made.

2 Academic degree organization at UFP

University Fernando Pessoa (UFP) adopted the 3+2 Bologna model for all degrees in 2006, except for degrees such as those in Health Sciences and Architecture where the Portuguese Ministry of Higher Education or European recommendations stated otherwise. The first cycle or undergraduate studies lasts at least 3 years and it's expected to train students to enter the labor market. The 2nd cycle or graduate studies lasts typically 2 years and corresponds to master-level studies. An academic year has 60 ETCS[1], and so a full 3-year undergraduate degree has 180 ECTS.

In the Bologna model, credits express the "quantity" of learning necessary to acquire the competences defined for that course. The learning outcomes of the course define the "content" of that learning. Learning outcomes define the competences that the student is expected to develop. Learning outcomes are statements of what a student is expected to know, understand and be able to demonstrate to be awarded the credits. Competences represent a combination of knowledge, understanding, skills and abilities. The final mark of the course grades the level of competences developed by the student.

The science and engineering courses at UFP have a three year first cycle, and a two year second cycle. An academic year at UFP has two terms, typically 18 weeks each. The first term runs from September to January, and the second term from February to June. A course offering spans only one term.

The transition from the 5-year Engenharia Informática degree to the current 3-year degree was based mainly in reducing the number of contact hours of the courses. That effort was also used by UFP to revise the overall plan, in line with international recommendations, such as those of the Association of Computing Machinery and of the Institute of Electrical and Electronics Engineers (ACM/IEEE). The next section describes the relationship between the Engenharia Informática and the ACM/IEEE recommendations.

The Engenharia Informática degree attracts typically 30 new students each year, having had stable inputs and outputs for the past years. There are classes during the normal working

[1] An ECTS (European Credit Transfer System) corresponds to a unit of student work of 26 hours. This includes lectures, self-study, assessment activities, field work, projects, and in general other learning activities. The number of hours varies from country to country, ranging from 25 to 30 hours.

hours and night classes (after 18h00) for working students. More and more students prefer after office hours classes, as more and more students seek income sources to finance their studies. The ratio of day to night classes' size is 2 to 1. Night students typically are not enrolled in a full term, as they do not have the time required to work for all courses.

2.1 The Engenharia Informática degree

The Engenharia Informática degree was proposed according to the national recommendations for engineering courses, and also according to the international studies and recommendations for courses in the area of computing. The former recommendations include the ACM/IEEE "Computing curricula proposal" (ACM/IEEE CC, 2005) and the ACM/IEEE "Computer Science Curriculum" (ACM/IEEE CSC, 2008), briefly described below.

The ACM/IEEE computing curricula proposal (ACM/IEEE CC, 2005) identifies five general academic fields in computing:

- Computer Engineering
- Computer Science
- Information Systems
- Information Technology
- Software Engineering

Due to the different approaches to the definition of degrees in this area in the US and in Portugal (and in Europe in general), there is not from the list above a single match for the Engenharia Informática degree. Nevertheless, by carefully analyzing the body of knowledge proposed by the ACM/IEEE task force, the Engenharia Informática degree at University Fernando Pessoa best matches the Computer Science curriculum as we will see in the next paragraphs.

Computer Science is defined as a field that spans the range from theory through programming, and which provides graduates with distinctive skills to perform the following broad tasks (ACM/IEEE CC, 2005):

- Design and implement software.

- Devise new ways to use computers.
- Develop effective ways to solve computing problems.

In trying to make further a distinction with the other fields, the document states that "*They design and develop all types of software from systems infrastructure (operating systems, communications programs, etc.) to application technologies (web browsers, databases, search engines, etc.). Computer scientists create these capabilities, but they do not manage the deployment of them*". This definition can be used to characterize UFP's Engenharia Informática graduates. Another often used translation by some Engenharia Informática degrees in Portugal is Computer Engineering, defined by the ACM/IEEE in (ACM/IEEE CE, 2004) as "*(…) a discipline that embodies the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-controlled equipment. Computer engineering has traditionally been viewed as a combination of both computer science (CS) and electrical engineering (EE)"*.

The Engenharia Informática degree trades a more electrical engineering focus of Computer Engineering for the more theoretical, foundational based approach of Computer Science. The Engenharia Informática degree does not offer extensive courses on Circuits and Signals, Electronics, Digital Signal Processing, or VLSI Design and Fabrication as required by Computer Engineering. It seems thus reasonable to compare the Engenharia Informática degree with the Computer Science curriculum recommendations.

In the rest of the document, unless stated otherwise, the Portuguese name of *Engenharia Informática* will be used.

The ACM/IEEE "Computer Science Curriculum" identifies 14 core knowledge areas (ACM/IEEE CSC, 2008) and 2 additional areas for consideration. The next table shows those 14 core knowledge areas and the 2 additional areas, which are shown in the last row. When such areas exist in the Engenharia Informática degree, there is an indication of the corresponding topics and in which cycles (1^{st} or 2^{nd}) they are offered; courses covering the topics from a given area can be offered in both cycles.

core knowledge areas	
Discrete Structures *(1st cycle)* Basic Logic Graphs and Trees	Human-Computer Interaction *(2nd cycle)* Foundations Multimedia and Multimodal systems Collaboration and Communication
Programming fundamentals *(1st cycle)* Data Structures Algorithms Object-oriented	Graphics and Visual Computing *(2nd cycle)* Computer Vision
Algorithms and Complexity *(1st cycle)* Fundamental Algorithms	Intelligent Systems *(2nd cycle)* Basic Search Strategies Agents Planning systems
Architecture and Organization *(1st cycle)* Computer Architecture Digital Logic	Information Management *(1st, 2nd cycles)* Information Models Database Systems Data Modeling Indexing Relational database design
Operating Systems *(1st cycle)* Operating systems Security Models	Social and Professional Issues *(2nd cycle)* Professional Ethics Computer Crime
Net-centric Computing *(1st, 2nd cycles)* Network Communication Mobile Computing Compression Multimedia Technologies	Software Engineering *(2nd cycle)* Software Design Using APIs Component Based Computing
Programming Languages *(1st, 2nd cycles)* Object-oriented programming Functional programming Declarations and Types	Computational Science *(1st cycle)* Operations Research
additional areas	
Introduction to Parallelism	Introduction to Security (2nd cycle) Introduction to Computer Security

Table 1: ACM/IEEE Computer Science Knowledge Areas

As shown in Table 1, the Engenharia Informática degree at UFP offers several courses covering the topics from the Knowledge Areas of the ACM/IEEE Computer Science Curriculum 2008. The degree offers also several courses from other areas such as Language Skills, Mathematics, Physics, and Management. There are also elective courses the students can choose from any degree in the Faculty of Sciences and Technology.

The next table shows the courses offered in the first cycle of the UFP's Engenharia Informática degree[2]:

1st year	2nd year	3rd year
Introduction to client-side web programming (4)	Numerical Analysis (5)	Foreign Language (4)
Physics (5)	Algorithms and Data Structures I (6)	Computer Networks I (7)
Electromagnetism (8)	Operations Research (4)	Multimedia I (7)
Mathematics I (7)	Programming Languages I (7)	System Analysis (7)
Information Systems (6)	Computer Architecture (6)	Database Systems (5)
English (4)	Digital Systems (6)	Portuguese and European Political Organization (4)
Applied Statistics (7)	Mathematics II (8)	Management (4)
Algorithms and Programming (7)	Algorithms and Data Structures II (6)	Multimedia II (6)
Elective course (4)	Programming Languages II (6)	Elective course (4)
Electronics (8)	Operating Systems (6)	Computer Networks II (6)
		Software Engineering (6)

Table 2: UFP's Engenharia Informática 1st cycle degree.

The second cycle offers courses from two majors:

- Information Systems and Multimedia.
- Mobile Computing.

Students can graduate in either of these majors by taking the corresponding specific courses, which are in the number of four; all other courses are offered to both majors.

The next table shows the courses offered to the two majors. Following the course name we show the number of ECTS. When there are two course names in the same cell, the first choice is from "Information Systems and Multimedia" and the second choice is from the "Mobile Computing" major.

[2] The number of ECTS follows the course name.

2nd cycle, 1st year	2nd cycle, 2nd year
IS Planning and Development / Mobile Networks and Services I (6)	Final Project I (12)
Mobile Computing (6)	Knowledge Management / Mobile Networks and Services II (3)
Professional Ethics (2)	Multimedia and Interactive Systems / Mobile Applications Programming (4)
Academic Internship (8)	Programming Paradigms / Mobile Applications Project (4)
Advanced Database Systems (6)	Elective course (4)
Artificial Intelligence (8)	Research Methods (3)
Distributed Systems (8)	Final Project II (3)
Man-Machine Interaction (6)	Dissertation (27)
Computer Security and Auditing (6)	
Elective course (4)	

Table 3: UFP's Computer Science 2nd cycle degree.

As we can see from the table, almost half of the 2nd cycle credits are for internships, final project, elective courses, and the dissertation work (making a total of 49 ETCTS). The other half of the credits are for Engenharia Informática-specific courses.

The Advanced Database Systems course is offered in the first year, second term of the 2nd degree and it follows a more general Database Management Systems course in the 1st degree. Although students are not required to take them first, other courses provide inputs to the Advanced Database Systems course. The Information Systems course provides fundamental concepts, and some techniques for conceptual modeling, e.g., the Entity-Relationship and the Object-Oriented models. The Operating Systems course introduces some other concepts and techniques, as memory management, buffer management, file organization and paging. The programming languages courses offer the skills for programming database access and manipulation code other than from the command line or graphical client interfaces.

In the next section we describe the Advanced Database Systems course.

3 The Advanced Database Systems course

The Advanced Database Systems course is offered in the 4th year of a 5 year Engenharia Informática degree in the 3+2 Bologna format — the 4th year is the first year of the second cycle. Students taking the Advanced Database Systems course are expected to have a good theoretical and working knowledge of Database Management Systems, Operating Systems, and Algorithms and Programming Languages. If they have completed a 1st cycle in Engenharia Informática, Computer Science, Computer Engineering, Software Engineering, Information Systems or a similar 1st cycle developing the same skills they should have all necessary background to take the course.

Students are expected to have taken a Database Management Systems course that exposed them to the fundamentals of databases. The introductory Database Management Systems course belongs to the Information Management (IM) area of knowledge in the ACM/IEEE Computer Science curricula. The broad topics covered in the introductory Database Management Systems course at UFP are:

E-R modeling	Normalization	Transactions
Relational Model	SQL	Concurrency control
Logical design	Storage and Indexing	Recovery

Table 4: Database Management Systems topics.

The labs section overs also databases and the web, introducing important issues when using databases, such as connection pooling, changing isolation levels, working with counters, defining triggers and views, and doing application level logging. These issues are dealt with from a programmer's point of view only, and some are only proposed as additional work, leaving to the students the choice to test and implement them or not.

The ACM/IEEE "Computer Science Curriculum" Information Management area does not explicitly propose topics for an advanced database systems course. However, some of the topics in the Information Management area of the curriculum can be introduced in the Advanced Database Systems course, for example:

- Distributed Databases.
- Data Mining.

- Digital Libraries.
- Information Storage and Retrieval.

As of the current offering, none of these topics is dealt with in the UFP's Advanced Database Systems course. Another effort to define a Database Body of Knowledge, the DBTech Pro project in Europe, aimed at, among others, "the identification of professional roles and skills in relation to contemporary database technology practice in Europe". The DB Tech project defined several courses, at beginner, intermediate and professional level, with feedback from some 150 companies throughout Europe. Most of the topics in the "Intermediate Knowledge" (IK) and "Data Access" (DA) courses are also present in the Advanced Database Systems course (DBTEchNet, 2012). These topics are:

- IK.12.01, Concurrency control (locking and versioning)
- IK.12.02, Backup and Recovery Techniques
- IK.14 Object-Relational DBMS
- DA.05.01 Concurrency Control (Application Level)
- DA.05.02 Concurrency Conflicts (Application Level)

The DBTEchNet proposed also a Database Administration course, but this level of skills is not currently provided by the database courses at UFP. Students do database administration, but the practical aspects of load balancing, transaction analysis, security and access control, maintenance scheduling, backups, and other topics are not covered at that level of detail.

3.1 Course structure

The Advanced Database Systems course has 6 ECTS for a total workload of 156 hours, divided as follows: 18 hours lectures, 36 hours labs, and 102 hours of independent work which can be used for assignments, projects, research, evaluation and self-study. Students can use the Office hours to discuss problems, and get guidance and help. More often contact is made online, via email, the class forum, or Skype.

The Advanced Database Systems course extends the topics of the introductory Database Management Systems course, and introduces new ones. In the Database Management

Systems course there is no time to go into detail of some algorithms and to present a larger set of techniques. For example, concurrency control in an introductory course remains largely in the domain of locking. Likewise, recovery is introduced, but there is generally no time to go into the details and optimizations of industrial strength algorithms. Storage organization is introduced, but the different alternatives are not discussed in detail. That's where the Advanced Database Systems starts and goes to develop further those topics.

The topics which are reviewed and extended, including discussion of internal workings and algorithms, are:

- Transaction management.
- Concurrency control.
- Buffer and Memory management.
- Storage organization.
- Transaction and Restart Recovery.

The topics which are new and never previously dealt with in the Engenharia Informática degree from a database perspective are:

- The Object-Relational approach.
- The Object-oriented model of data.
- Object-Oriented Databases (OODB), Object Query languages.
- Representation of Spatial data.
- Spatial data operators and indexing.
- Non-SQL databases.
- Architecture and applications of columnar databases.

Some of the more advanced topics can eventually change from one year to the other. For example, active databases, temporal databases, logic and deductive databases and data warehouses are often part of similar syllabus in other institutions. We chose to introduce

object databases, spatial database extensions, and columnar databases due mainly to their theoretical and growing practical significance. However, it can be expected that spatial databases can be replaced on some academic terms by other multidimensional databases, for example for data mining or business intelligence purposes.

3.2 Course goals

The course goals are the following:

- To provide the students with a better understanding of the essential techniques used in a Database Management System, either by revisiting them or by studying new approaches.
- To provide students with knowledge to choose, design, and implement a database management system in a complex domain, making the best use of the available tools and techniques.
- To provide students with knowledge to analyze and tune a given database management system, given a workload and usage patterns.
- To allow the students to learn and experiment advanced database techniques, models and products, and to provide them with the knowledge to take decisions concerning implementation issues.
- To provide students with knowledge to analyze, modify if necessary and experiment algorithms that make up the database internals.
- To expose students to advanced topics and techniques that appear promising research directions.

The main course outcomes, in terms of what students are expected to be able to perform upon successful completion of the course, are the following:

1. Describe database management system internals. Understand and describe internal algorithms in detail.
2. Identify and be able to use recent and advanced database techniques (e.g. in concurrency control, buffer management, and recovery).

3. Decide on configuration issues related to database operation and performance. Identify which parameters are tunable and what are the implications.

4. Analyze and optimize transactional code, identifying causes of possible anomalies and correct them.

5. Decide on optimization issues given a known database workload, by manipulating indexes, choosing more adequate data types, and modifying queries.

6. Identify limitations of the standard Relational databases in certain application domains, e.g. for multidimensional data, or unstructured data.

7. Analyze, describe and use other models than the Relational.

8. Identify opportunities for the use of the object model, and design and code client code to manipulate an object database.

9. Analyze, compare and evaluate alternative database architectures and models in different application contexts.

Furthermore, being an engineering degree, it is expected that students develop also the following skills (Irvine, 1998). Most of those skills are to be acquired during the whole degree, and not exclusively in a given course:

- Ability to apply knowledge of mathematics, science and engineering.
- Ability to design and run experiments, and to analyze the results.
- Ability to design and implement a system or component to satisfy a given set of goals.
- Ability to work in multidisciplinary teams.
- Ability to identify, to define, and solve engineering problems.
- Knowledge and understanding of professional and ethical issues.
- Ability to communicate.
- Acquire a broad education necessary to understand the implications of engineering solutions in a social and larger context.

- Awareness of the need for lifelong learning.
- Knowledge of contemporary issues.
- Ability to use the engineering techniques, tools and skills for their daily professional life.

The Advanced Database Systems course contributes partially to some if not to all of these outcomes. We give some examples below:

Ability to design and run experiments, and to analyze the results.

Students are proposed case studies to analyze and to define how to test them. For example, the test of concurrency problems is often difficult, and students should understand what is being measured, why, and how.

Knowledge and understanding of professional and ethical issues.

Database applications and the implications of large-scale database use are discussed in some of the examples. A database administrator and a database programmer have access to potentially sensitive, private information which needs to be protected from a professional, legal and ethical perspective.

Awareness of the need for lifelong learning.

Students are faced with the changing nature of the field, in a fast pace in some domains, and should become aware of the need to keep constantly updated. For example, the last 10 years have seen a rapid growth and offer of so-called non-SQL databases, and of their increasing acceptance in some domains (like business intelligence and eScience).

Knowledge of contemporary issues.

Databases are used to store and process sensitive information, and students should be aware of the legal, ethical and societal factors involved. Databases are used to record all kinds of information, anytime, and can be hosted anywhere in the world subject to different laws, regulations and procedures.

Ability to use the engineering techniques, tools and skills for their daily professional life.

Throughout the study of the internals of the database management system, students have the possibility to discuss techniques and tools they may have used elsewhere. Also, students are reminded of real world applications of databases and of some other tools used in the course.

3.3 Course syllabus

The course syllabus is organized in 3 main sequential parts, corresponding to a total of 54 contact hours (lectures and labs) during the academic term, divided as follows:

module	topics	Contact hours / ECTS
Database internals and advanced algorithms	▪ Transaction Management ▪ Concurrency control ▪ Storage organization ▪ Recovery	24 hours / 2,5 ECTS
Object-oriented databases	▪ The Object-Relational approach ▪ Object-oriented databases ▪ Object Query languages ▪ Architecture of a OODB. Transactions.	12 hours / 1,5 ECTS
Other database models	▪ Representation of Spatial data ▪ Spatial data operators and indexing ▪ Non-SQL databases ▪ Architecture and applications of columnar databases	18 hours / 2 ECTS

Table 5: Course syllabus.

Those topics are covered in the lectures and in the labs. The lectures and the labs follow the plan described in the next section.

3.4 Lecture plan

The lecture plan for an average 18 weeks term is the following. Each weekly topic corresponds typically to a 1 hour class and a 2 hour lab session:

1. Introduction. The database canonical architecture. Components to review.
2. Concurrency Control. Locking, Optimistic and Timestamps algorithms.
3. Multi-version Concurrency Control. New Isolation levels. Multi-version serializability.
4. Snapshot Isolation (SI). SI Anomalies. Applications of SI. Products using SI.
5. Models of Storage organization.
6. Concurrency control and recovery, logging.
7. Transaction and restart recovery, the ARIES algorithm. Point in Time Recovery.
8. First written exam.
9. The Object-Oriented Model. Applications for non-structured data. Approaches.
10. Object-Oriented Databases, extensions to Programming Languages.
11. Queries in Object Databases.
12. Transactions in Object Databases. Optimistic and Pessimistic algorithms.
13. Multidimensional data, the Geometric model, PostgreSQL examples.
14. Geometric Indexing. The GiST index structures. PostGIS.
15. Non-relational models. The columnar stores. Architecture of MonetDB.
16. Architecture of column stores. Other examples (Bigtable, C-Store, Cassandra, HBase, Triplestores, MonetDB).
17. Queries in column stores. The examples of Big Data and eScience.
18. Second written exam.

3.5 Course contents

This section describes in detail the course contents, the supporting materials used, resources, and the main bibliography. Where books are mentioned, there are references to the chapters that are used.

3.5.1 First module

The first module spans over the first 7 sessions. It reviews important database introductory concepts, and extends them with new techniques and approaches. This module deals primarily with concurrency control and recovery, revising the lock mechanisms and introducing the Timestamp Ordering, the Optimistic, the Multi-version, and the Snapshot Isolation algorithms and protocols.

The relationship between memory management and concurrency control and recovery is briefly discussed. Recovery approaches are briefly revised, and the ARIES algorithm is presented in detail.

Finally the module presents storage organization and discusses current trends, emphasizing cache-conscious and memory-conscious approaches. This material will be used later when the columnar architectures are introduced.

Session 1

The first session introduces the course. The lecture plan is presented and detailed, as the grading method. The course goals are described as the expected outcomes. The bibliography and the software to be used are also presented. A brief overview of the standard database architecture is given, with emphasis on the modules to be detailed in the first module.

In the labs, students are encouraged to install, or to upgrade to, PostgreSQL 9.x. A brief overview of the documentation is given, as of the main characteristics of the system. Students should also install an IDE, typically Eclipse or NetBeans.

Topics:

> Introduction. Course contents. Description of goals and outcomes. Rules for assignments, exams and grading. Rules for participation in classes. Attendance. Brief

overview of the classic database architecture, emphasizing the components to be studied in the course. Introduction to the first module.

Methods:

Lecture.

Bibliography:

- Course notes, course lecture plan. Available online in the course area.

Recommended readings:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapters on Transactions and Concurrency Control.
- Bernstein, P. A., V. Hadzilacos, N. Goodman (1987). **Concurrency Control and Recovery in Database Systems**, Addison-Wesley. Chapter 1.
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 6, 7.

Session 2

This session starts by a brief overview of concurrency control, revisiting locking algorithms. It then introduces non-locking approaches, such as Optimistic or Certification algorithms, and Timestamp Ordering algorithms. Students should understand the importance of the scheduler, and how it can serialize schedules under different policies. Students are also exposed to research issues, such as concurrency control policies and applications in distributed systems, and in mobile systems. Pointers to other less known algorithms are given, showing students the field is open to contributions and to innovation. Classes of schedules under serialization and recovery policies.

Topics:

Concurrency Control. Locking, Optimistic and Timestamp Ordering algorithms. The algorithms, examples. Brief performance comparison. Schedules produced by different algorithms. Other concurrency control algorithms. .

Methods:

Lectures and quizzes. Students should work out several schedules, by hand, and compare the different approaches. No software is used as these algorithms are not implemented on mainstream products.

Bibliography:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapters on Transactions and Concurrency Control.

Recommended readings:

- Bernstein, P. A., V. Hadzilacos, N. Goodman (1987). **Concurrency Control and Recovery in Database Systems**, Addison-Wesley. Chapters 2, 3 and 4.
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 16, 17.

Assignments

- Problem set #1: exercises about serialization, and locking, timestamp ordering and optimistic schedulers.

Session 3

This session introduces Multi-version Concurrency Control (MVCC) as an alternative to the performance issues of the previous approaches. It is shown that read-intensive applications can see huge performance improvements. Starting from the distributed systems approach of the first MV proposals, the variants with Timestamp Ordering (MVTO) and 2Phase locking (MV2PL) are introduced. Students are then exposed to the practical complexities of a MV scheduler.

Topics:

Multi-version Concurrency Control (MVCC). MVTO and MV2PL. Why MVCC is attractive. Multi-version schedules. How to test for serializability. What products implement MVCC. The cost of MVCC.

Methods:

Lectures and quizzes. Students solve several MVCC schedules, by hand. First tests with MVCC in PostgreSQL.

Bibliography:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapters on Transactions, Concurrency Control, Memory Management and Storage Organization.

Recommended readings:

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- Bernstein, P. A., V. Hadzilacos, N. Goodman (1987). **Concurrency Control and Recovery in Database Systems**, Addison-Wesley. Chapters 3 and 4.
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 16, 17.

Session 4

This session introduces the practical implementations of MVCC, the so called Snapshot Isolation. It is shown that Snapshot Isolation creates an isolation level of its own, and offers a practical and efficient solution to the MVCC protocol. Then, the First Committer Wins and the First Updater Wins variants are discussed. The importance of SI is shown by discussing the commercial and open source products that have adopted it. An IBM research paper, arguing against the Oracle approach to SI allows students to read different positions, and to make up their own mind. Also, it shows them that several factors decide the life of the techniques, regardless of their technical attributes. Students should be able to reproduce SI anomalies and understand the importance of choosing the right isolation level and of defensive coding against common anomalies. They are also shown that standards get implemented differently by different products.

Topics:

Snapshot Isolation (SI). New isolation levels. Products using SI: SI implementation. So-called Read Consistency implementations. Snapshot Isolation Anomalies. Serializable SI. SI performance.

Methods:

Lectures and quizzes to be solved with PostgreSQL. Students are given a couple of papers to analyze, including the IBM paper (IBM 2002) and a SI paper (Alomari *et al* 2009).

Bibliography:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapters on Transactions, Concurrency Control, Memory Management and Storage Organization.

Recommended readings:

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- IBM (2002). **A Technical Discussion of Multi Version Read Consistency**, IBM Software Group, Toronto Laboratory, August 2002, 13 p.
- Alomari, M., A. Fekete et al (2009). **A Robust Technique to Ensure Serializable Executions with Snapshot Isolation DBMS**, IEEE International Conference on Data Engineering, Xangai, China.
- Bernstein, P. A., V. Hadzilacos, N. Goodman (1987). **Concurrency Control and Recovery in Database Systems**, Addison-Wesley. Chapters 3 and 4.
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 16, 17.

Assignments

- Problem set #2: exercises about serialization, and multi-version schedulers. Exercises with PostgreSQL serializable snapshot isolation level.

Session 5

This session presents and discusses the Storage Manager, the component that organizes the database in the disk. Different storage models are presented, and analyzed in different usage scenarios. The implications of storage models in performance are discussed. The pros and cons of each model are discussed.

Topics:

Models of Storage organization. N-ary Storage Model. Decomposition Storage Model. Partition Attributes across Model. Clotho Model.

Methods:

Lecture, analysis.

Bibliography:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapter on Storage Organization.

Recommended readings:

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 8, 9.

Session 6

This session presents the Buffer Manager and its relation to the Transaction Manager. The memory management algorithm used by PostgreSQL is presented and discussed. Several other buffer management algorithms are presented including MySQL's and Oracle's. The relationship between memory management and recovery is discussed and several alternatives are presented. Factors affecting the performance of the algorithms are discussed. The relation of the Buffer manager to recovery procedures is discussed. The policies resulting from this relationship are discussed.

Topics:

The Buffer Manager. Memory management, relation to concurrency control and recovery. Page replacement algorithms. The relation between the Buffer Manager and recovery. Recovery procedures. Shadow Paging and Logging.

Methods:

Lectures. Quizzes about the different page replacement algorithms. Analysis of the PostgreSQL Buffer Manager implementation.

Bibliography:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapters on Memory Management, and Recovery.

Recommended readings:

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- PostgreSQL 9.1 source code, available at `http://www.postgresql.org/developer/coding/`
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 17, 18.
- Garcia-Molina, H., J. Ullman *et al* (2008). **Database Systems: The Complete Book**, Prentice Hall. Chapters 17, 19.

Assignments

- Problem set #3: exercises with page replacement algorithms. Pencil and paper simulation, and comparison of algorithm implementations.

Session 7

This session discusses further the Recovery Manager, and introduces the WAL principle. It is shown that WAL can provide the redundancy necessary to ensure recovery. The logging approach is further discussed, and a specific implementation of WAL, the ARIES algorithm,

is presented. ARIES is discussed for transaction recovery and restart recovery. Specific applications such as Point in Time Recovery are also discussed.

Topics:

The Recovery Manager. Interaction between the RM, the Buffer Manager and the Transaction Manager. Logging strategies. WAL. The ARIES algorithm. ARIES data structures and passes. Hot standby with Point in Time Recovery (PITR).

Methods:

Lectures and labs with PostgreSQL.

Bibliography:

- Gouveia, F. "Bases de Dados". Draft notes for the database book. Chapter on Recovery.
- Mohan, C. D. Haderle *et al* (1992). "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", **ACM Transactions on Database Systems 17**(1).

Recommended readings:

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapters 17, 18.
- Garcia-Molina, H., J. Ullman *et al* (2008). **Database Systems: The Complete Book**, Prentice Hall. Chapters 17, 19.

Assignments

- Problem set #4: exercises about recovery with the ARIEs algorithm.

Session 8

First written exam.

3.5.2 Second Module

The second module introduces new database models that provide alternative solutions to some of the problems of the Relational model when dealing with complex, structured data. The model chosen is the Object-Oriented model, for the following reasons:

- It provides a seamless integration with the popular object-oriented languages, such as Java, C++ or C#.
- It is a well understood model, with a good deal of standardization and existing prototypes and products since the late 80s.

This module starts with the so called object-relational variation, and then introduces object-oriented databases. It discusses the different approaches to Object-Oriented databases, and proposes a particular database to study further. Topics such as object organization in disk, memory management, transactions and query languages are presented. Emphasis is placed on the low "impedance mismatch" factor of these databases.

The second module starts after the first written exam, and spans the following sessions.

Session 9

This session introduces the Object-Relational approach as a viable alternative to problems where the tabular structure of the Relational Model is less adequate. Examples are taxonomies, graph models, semantic models, and CAD models. The decompositions of the object model are presented, and the particular PostgreSQL approach to inheritance is presented. This module uses PostgreSQL 9.x to work with an object-relational layer. PostgreSQL is chosen because the object-relational capabilities are embedded in the system since the beginning. As such, they are clear to understand. Further, meta-model approaches are shown to be flexible and of practical importance.

Topics:

> Applications for non-structured data. The Object-Relational approach. Decomposition methods for object models. Horizontal, Vertical and mixed decomposition. Performance considerations. The PostgreSQL inheritance model. Meta-models.

Methods:

Lectures and labs with PostgreSQL. Exercises about object decomposition.

Bibliography:

- Gray, P., K.G. Kulkarni, and N.W. Paton (1992). **Object-Oriented Databases: A Semantic Data Model Approach**. Prentice-Hall, Hertfordshire.
- Blaha, M., Premerlani, W. (1998). **Object-Oriented Modeling and Design for Database Applications**, Prentice-Hall. Chapter 13.
- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`

Recommended readings:

- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapter 23.

Assignments

- Problem set #5: simple design using the O-R features of PostgreSQL. Typically the University Student System problem is used.

Session 10

This session presents Object-Oriented databases (OODB), their approaches and architecture. OODB are classified according to persistence approaches. Examples of OODB approaches are given. To experiment with an OODB we chose **db4o**, by Versant, which is one of the pioneers in the Object-Oriented databases world. **db4o** offers a Java and a .NET interface, it's simple to install and setup, and offers most of the functionalities discussed in this module.

Topics:

The Object-Oriented Model. Object-Oriented Databases, extensions to Programming Languages. Persistent libraries. Object identity. Architectures. Introduction to **db4o**.

Methods:

Lectures and labs with db4o.

Bibliography:

- db4o documentation, available at `http://www.db4o.com`
- Chaudhri, A., Mary Loomis (1998), **Object Databases in Practice**, Prentice-Hall. Chapters 1, 2, 14.

Recommended readings:

- Kim, W. (1990). *Object-Oriented Databases: Definition and Research Directions*, **IEEE Transactions on Knowledge and Data Engineering, 2**(3).
- Greene R. (2006).*OODBMS Architectures*, Versant Corporation.
- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapter 23.

Assignments

- Problem set #6: research, implementation and presentation of a topic.

Session 11

This session introduces Object Query Languages, and the Object-oriented extensions to the SQL language. Query by Example, and Native Queries are discussed.

Topics:

Queries in Object-oriented Databases. Query by Example. Native Queries. The **db4o** approach.

Methods:

Lectures and labs (Tutorial) with db4o.

Bibliography:

- db4o documentation, available at `http://www.db4o.com`
- db4o Tutorial for java, available at `http://www.db4o.com`

Recommended readings:

- Chaudhri, A., Mary Loomis (1998), **Object Databases in Practice**, Prentice-Hall. Chapters 2, 14.

Assignments

- Problem set #7: coding of the db4o tutorial. Students are informed from the beginning in session 10 that they should submit their testing with db4o as an assignment.

Session 12

This session introduces transaction and concurrency control approaches in OODB. Issues specific to the navigational aspect of these databases are discussed. The optimistic approach of db4o is discussed.

Topics:

Transactions in Object-oriented Databases. Concurrency Control in OODB. The **db4o** approach.

Methods:

Lectures and labs with db4o.

Bibliography:

- db4o documentation, available at `http://www.db4o.com`
- db4o Tutorial for java, available at `http://www.db4o.com`

Recommended readings:

- Chaudhri, A., Mary Loomis (1998), **Object Databases in Practice**, Prentice-Hall. Chapter 14.

- Fabrice Lapie, Feliz Ribeiro Gouveia, “Performance Analysis of three OODB”, OOSE Project, Institut International pour l’Intelligence Artificielle, Centre de Transfert da Université de Tecnologie de Compiègne, France, July 1993.

3.5.3 Third Module

The third and last module introduces two other database models: the spatial model and the columnar, non-SQL model. The spatial model is important because more and more systems store geographic and/or geometric information. Geographic Information Systems, location-aware systems, GPS are just some of the examples. Traditional relational databases cannot efficiently cope with multidimensional data. The spatial data model, as most of the DBMS now offer, provides a geometric data model and queries over that model. Students can experiment with the PostgreSQL geometric extension, and study the PostGIS layer.

The last part of the third module presents non-SQL models, giving as example the so-called columnar databases. Examples of Google, Facebook are given to understand the nature of data that does not fit in the relational model. Triples, or variants thereof, are used to demonstrate the need for other models. The students have the opportunity to experiment with MonetDB, a column store that is SQL-compliant and offers a complete set of API to most common languages. MonetDB is an open-source columnar database developed at the **Centrum Wiskunde & Informatica** (CWI), in the Netherlands.

Session 13

This session introduces multidimensional data and discusses the issues of indexing and querying that data. Examples are given for geometric and geographic databases. In-memory and disk based indexing structures are presented. The PostgreSQL approach to multidimensional data is presented.

Topics:

> Multidimensional data. Indexing approaches: symmetric and non-symmetric structures. R-Trees. The Grid File. The Generalized Index Structure: GiST. Spatial data in PostgreSQL. Examples with PostGIS.

Methods:

> Lectures and labs with PostgreSQL.

Bibliography

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- PostGIS PostgreSQL Spatial Extension, available at `http://postgis.refractions.net/.`

Recommended readings:

- Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill. Chapter 28.

Session 14

This session uses the PostgreSQL spatial data types and uses the indexing capabilities. An introduction is made to PostGIS which is a PostgreSQL module with more advanced geometric capabilities. PostGIS is added to allow PostgreSQL spatial capabilities to be used in Geographic Information Systems. Due to lack of time PostGIS is generally not installed and tested, but students are invited to do it if they find time.

Topics:

Spatial data in PostgreSQL. The geometric data types. GiST indexing. Examples with PostGIS.

Methods:

Lectures and labs with PostgreSQL.

Bibliography

- PostgreSQL 9.1 documentation, available at `http://www.postgresql.org/docs/9.1/static/index.html`
- PostGIS PostgreSQL Spatial Extension, available at `http://postgis.refractions.net/.`

Recommended readings:

- Blasby, D. “Building a Spatial Database in PostgreSQL”, presentation, Refractions Research.

Assignments

- Problem set #8: coding of a model with spatial information in PostgreSQL. Queries with spatial restrictions.

Session 15

This session introduces applications where relational database systems are thought to behave poorly in terms of performance. Examples include the semantic web and triples, and typical data warehouse applications. The architecture of column-oriented databases is introduced. Data representation in columns is compared to the line-oriented tables. RDF databases and triplestores are introduced. Introduction to MonetDB.

Topics:

RDF databases. Triplestores. Columnar databases. Column-oriented architectures. Storage and indexing in column-oriented stores.

Methods:

Lectures and labs with MonetDB.

Bibliography

- MonetDB, available at `http://www.monetdb.org/`

Recommended readings:

- Abadi, D., S. Madden, N. Hachem (2008). “Column-Stores vs. Row-Stores: How Different Are They Really?”, SIGMOD’08, June 9–12, Vancouver, Canada.

- Victor Magalhães, Nuno Fernandes, Feliz Ribeiro Gouveia (2007). “Managing large datastores using RDF databases: an experiment in event tracking”, Proceedings of 2nd *Conferência Ibérica de Sistemas e Tecnologias de Informação (CISTI 2007)*, Vol II,

pp. 173-181, ISBN: 978-972-8830-88-5, 21-23 June, UFP, Edições da UFP, Porto, Portugal.

- DeCandia, G. *et al* (2007). "Dynamo: Amazon's Highly Available Key-value Store", Proceedings of ACM SOSP'07, October 14–17.

Session 16

Architecture of Column-oriented databases (C-stores). Differences between C-stores and relational databases. Storage organization. Performance factors. Examples and analysis of benchmark code (TPC-H, SSB and derived benchmarks).

Topics:

Architecture of Columnar databases. Storage organization models. Query processing. Benchmarks.

Methods:

Lectures and labs with MonetDB.

Bibliography

- MonetDB, available at `http://www.monetdb.org/`

Recommended readings:

- Abadi, D., S. Madden, N. Hachem (2008). "Column-Stores vs. Row-Stores: How Different Are They Really?", SIGMOD'08, June 9–12, Vancouver, Canada.

- Transaction Processing Council, http://www.tpc.org/

Assignments

- Problem set #9: coding of a star model in MonetDB. Coding of a simple client.

Session 17

This session is dedicated to design issues in columnar databases and how they impact performance. A star-like schema is used to show how to deal with large sets of data. Example databases with tenths of millions of facts are used in the labs.

Topics:

Design in columnar databases. Performance issues in columnar databases. Discussion of examples. Running benchmarks.

Methods:

Lectures and labs with MonetDB.

Bibliography

- Harizopoulos, S., D. Abadi, P. Boncz (2009). "Column-Oriented Database Systems", Conference on Very Large Databases, VLDB, Tutorial, Lyon.
- MonetDB, available at `http://www.monetdb.org/`

Recommended readings:

- O'Neil, P., O'Neil, B., Chen, X. (2009). "Star Schema Benchmark, Revision 3".
- SSB source code, https://github.com/electrum/ssb-dbgen
- Ramakrishnan R. and J. Gehrke (2003). Database Management Systems, 3rd ed., McGraw-Hill. Chapter 25.
- Stonebraker *et al* (2005). "C-Store: A Column-Oriented DBMS", Conference on Very Large Databases, VLDB Conference, Trondheim, Norway.
- Chang, F. *et al* (2006). "Bigtable: A Distributed Storage System for Structured Data", Proceedings of OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA.

Assignments

- Problem set #10: coding of a benchmark model in MonetDB. Loading of a large data set (millions of data), typically from the Star Schema Benchmark.

Session 18

Second written exam.

4 Methodology

The course interleaves lectures with laboratory work. Lectures introduce the topics and go into detail of the more complex algorithms. In the labs students should test, code and analyze results. When possible, students are guided through the learning process in the labs by using tutorials, as is the case in module 2 using the db4o tutorial, a 160 page document detailing the programmer interface to the database and explaining the internals.

Students use their laptops, where they install and run the software. There is no database server the class can use. There is little emphasis on formal group work, as sections tend to be small, from 6 to 10 students. Students meet informally and discuss and exchange solutions involving more complex code. Nevertheless, the solutions to the assignments and projects are individual. At least for one of the projects students should prepare a 15 minute presentation and discuss it with colleagues. Depending on the number of students and on the number of weeks in the term, this can be done in one or two lab sessions.

4.1 Methodological approach

The following methods are used throughout the course:

- Presentation and discussion of a topic, as much as possible illustrated with real life examples. Lectures are typically 1 hour sessions.
- Presentation of solutions and approaches to technical issues. Technical, commercial, legal or social viewpoints can be introduced if applicable. Part of these presentations can be made in lab time, and students can use software.
- Lab work to selected exercises. Students can solve exercises in lab time, which can also be used to detail a technique or explain a topic.
- Recommended reading about the topics, typically from a historical or research (recent) perspective. Three to four papers are recommended during the course.
- Presentation by the students of at least one project, describing all the research, reading, and implementation done. Students are given generic topics (e.g. database clustering) and should prepare a demonstrator for that topic. They should also prepare themselves for Q&A following the presentation.

Most of the practical work happens in the beginning of the term as this implies coding, debugging and can be time consuming and with an unpredictable effort. Students have typically more time at the beginning, while at the end of the term they have heavier loads as all deadlines and exams tend to converge in the same week. So, it is better for students to do the heavy work in the beginning, and leave more theoretical work, or guided practical work, to the end.

The resources used include software, support material (slides, exercises, tutorials), and bibliography (book chapters and papers). Students use full-fledged database systems, currently three of them. While using 3 database systems could add to complexity with no real benefits to the students, we have found that students adapt well and understand the topics for each of them.

4.2 Research issues

Being a 2nd cycle, master level, course, it is expected that students be encouraged to use independent research techniques, and to be able to look for, analyze and discuss advanced materials.

Students are provided with journal papers and other scientific writings and they should be or become familiar with reading and analyzing this kind of works. For some of the sessions there are research papers as recommended reading. Students should understand some of the material they read describes theories, experiments, or both and can be subject to critique and to further improvement. There are several research opportunities in the field, and it is important that students realize the area is alive and moving forward.

4.3 Course resources

The course resources are essentially the class notes, quizzes, database software, and the bibliography (either at the libraries or at the **B-on** Digital Library). Students have access to the UFP learning management system, UFPUV, where all the material is posted.

The class notes are used to guide the students and to provide topics they can read further in the bibliography. They discuss also examples, annotated. Class notes are typically made available one week before the corresponding sessions.

Besides assignments, some topics need more extensive training, such as the serialization algorithms, in particular the multi-version cases. Additional sets of quizzes are made available, and can be discussed in the labs.

The course is based on the following open-source database management systems:

- PostgreSQL, an object-relational DBMS with geometric extensions (PostGIS)
- db4o, an object-oriented DBMS
- MonetDB, a column-store, non-relational, DBMS.

Depending on the students' preferences, those database management systems' Java, .NET, Perl or PHP API, where available, can be used. Students should be familiar with a development environment, typically Eclipse or NetBeans. All three database management systems are to be installed and run on the student's machines.

At least PostgreSQL is already known to the students coming from the 1st cycle's Database Management Systems course. Otherwise, it is a standards compliant product, easy to install and to start working with. PostgreSQL brings some advantages over other open source products such as MySQL. First, it offers full features of an industry-level DBMS, from the ground-up. MySQL offers them also, but some only after some configuration; for example, to use transactions, a transactional engine such as InnoDB must be chosen. PostgreSQL can also be used to learn and use the Object-Relational topics, and the spatial ones. Besides, the PostGIS extension to PostgreSQL is fully OpenGIS compliant.

As for the other DBMS they were chosen because they allow the students to learn and use the concepts, and they are also widely used and known. The OODB, db4o is simple to install and use, and lets students program directly inside the IDE. There is no added complexity in using the system and it is likely they'll have to use small footprint systems mainly in mobile applications. Having studied and used a database system that can rival Mini SQL (mSQL) and integrates seamlessly with Java is an added value. The last system, MonetDB is also very simple to install and use and students can find real use for a column-oriented database. More and more companies store huge amounts of data, and look for affordable, reliable, and viable alternatives to the relational model.

5 Assessment

The course assessment requires the students to attend at least 30% of the lectures, and 70% of the labs. The students can succeed in the lecture component, the labs component or both. If students succeed in both the lectures and the labs, they succeed in the course. Students failing the lecture component but satisfying the attendance requirement have a second chance in a final 2 hour exam at the end of the term. Students failing the labs component do not have a second chance and must enroll again in the labs section. If a student fails the labs but succeeds in the lectures, the lecture component mark remains valid for a period of at most 2 years.

The assessment of the lectures component is made of:

- Two written, open book, 2 hour exams;
- Class participation.

The assessment of the labs is made of:

- 7 to 10 assignments, which includes quizzes and coding projects;
- Lab participation, which can include presentations.

In both cases, the class and lab participation accounts for 10% of the mark. The final mark is the average of the final mark of each component.

A typical quiz should involve an average of 2 to 4 hours work. Students have typically 7 to 10 days to submit their answers or analysis. Students submit their work online.

Typical projects include reading, analysis and implementation, either by coding or configuration of existing products. Projects leave more freedom to choose and do research to students, whereas quizzes are more problem-solving assignments.

For module 1, typical projects include:

- “Point-in-time-Recovery” (PITR).
- Database replication techniques.
- Database clustering.

- Database recovery algorithms.

For module 2, typical projects include:

- Coding a “complex” problem in Java using the db4o API.
- Test transactions and queries controlling retrieval depth.

For module 3 typical projects include:

- Loading and querying large data sets.
- Setting up benchmarks with large data sets (SSB or TPC-H benchmark).

In module 1 most of the projects can use PostgreSQL even for advanced topics as replication and clustering.

In module 2 the projects use db4o and the Java or .NET API. Projects are coded in the IDE of choice of the students.

In module 3 the projects use MonetDB and the large datasets available.

6 Similar course offerings

Databases are a required topic in the computing field, and as such there are many offerings worldwide. To keep the course updated and in line with current trends, we regularly compare syllabi and methods with other institutions, especially those making the top in the field. The top 20 institutions by research publications in the field of Databases in the years 2000 to 2009 are:

- IBM, San Jose
- Microsoft, Redmond
- AT&T, Florham Park
- National University of Singapore
- University of Wisconsin, Madison
- Stanford University
- Hong Kong University of Science and Technology
- University of California, Berkeley
- University of Illinois, Urbana-Champaign
- University of Maryland, College Park
- University of Toronto
- University of Washington, Seattle
- Bell, Murray Hill
- University of Michigan, Ann Arbor
- Carnegie-Mellon University, Pittsburgh
- Purdue University, West Lafayette
- Chinese University of Hong Kong

- IBM, Yorktown Heights
- University of Waterloo
- University of California Riverside

The top European institutions by research publications in the field of Databases, in the same period, are:

- ETH Zurich, Switzerland
- Aalborg University, Denmark
- University of Edinburgh, Scotland
- INRIA Le Chesnay, France
- University of Athens, Greece
- Max Planck Institute Saarbrucken, Germany
- Centrum Wiskunde & Informatica (CWI) Amsterdam, The Netherlands
- University of Munich, Germany

Nonetheless, none of these institutions makes it to the world top 20. We describe the similar courses that are offered at some of the academic institutions in these lists, adding some not in the lists if the institutions are renowned for the excellence of its teaching and research in Computer Science. Courses are included if they have similar goals and outcomes, regardless of the name; for example, Database Systems courses are included because they are offered at graduate level, and bear similarity to our Advanced Databases course. Text copied verbatim from the course description is enclosed in quotes.

University of Wisconsin-Maddison

CS 764 Topics in Database Management Systems (Graduate level)

(`https://database.cs.wisc.edu/courses/`)

“This course covers a number of advanced topics in development of database management systems (DBMSs) and the application of DBMSs in modern applications. Topics discussed include advanced concurrency control and recovery techniques, query processing and

optimization strategies for relational database systems, advanced access methods, parallel and distributed database systems, extensible database systems, data analysis on large databases. The course material is drawn from a number of papers in the database literature, assigned at a rate of approximately two per week."

Stanford University

CS 245 Database Systems Principles (Senior/Graduate level)

(http://www.stanford.edu/class/cs245/)

"CS 245 is a senior/graduate-level introduction to the implementation of database management systems"

Topics include: Hardware (1 session), File and System Structure (2 sessions), Indexing (2 sessions), Hashing (2 sessions), Query Processing (3 sessions), Failure Recovery (2 sessions), Concurrency Control (2 sessions), Transaction Processing (2 sessions), Information Integration (1 session).

MIT

6.830/6.814 Database Systems (Graduate level)

(http://db.csail.mit.edu/6.830/sched.html)

"This course relies on primary readings from the database community to introduce graduate students to the foundations of database systems, focusing on basics such as the relational algebra and data model, schema normalization, query optimization, and transactions. It is designed for students who have taken 6.033 (or equivalent); no prior database experience is assumed though students who have taken an undergraduate course in databases are encouraged to attend."

Topics include: The Relational Model, Schema Design, Introduction to Database Internals, DB Operators and Query Processing, Indexing and Access Methods, Buffer Pool Design and Memory Management, Join Algorithms, Query Optimization, Transaction and Locking, Optimistic Concurrency Control, Recovery, Degrees of Consistency, C-Store, Distributed Transactions, Parallel DB, Scientific DB, NOSQL, ORM, Database as a Service.

University of California, Berkeley

CS 286 Implementation of Database Systems (Graduate level)

(https://sites.google.com/a/cs.berkeley.edu/cs286-sp09/)

"CS 286 is a graduate-level database course that covers the fundamentals of modern and emerging database and data-intensive systems architectures. The general approach is to focus on one aspect of the architecture at a time, and for each one, first catch up with the current state-of-the-practice, and then investigate emerging trends and future opportunities. Students are assumed to have already completed CS 262A - Advanced Topics in Computer Systems, which covers much of the foundational material from Systems and Database Systems. The bulk of the course is a set of readings along with a self-directed research project."

Topics include: DBMS Architecture Overview, Storage Hierarchies and Storage Management; Concurrency Control, Consistency, and Replication; Relational Query Processing (centralized, distributed, and parallel); Non-Relational Data Models and Processing; Stream, Temporal, and Continuous Processing; Dataspaces and Data Integration.

Carnegie Mellon University

15-823 Hot Topics in Database Systems (Graduate level)

(`http://www-2.cs.cmu.edu/~natassa/courses/15-823/`)

"The course covers performance issues in today's database system design. Topics include query processing and optimization, concurrency control, smart and efficient benchmarking, modern query processing algorithms for internet applications, interaction between the database software and the underlying hardware, and other topics related to database system performance."

Cornell University

CS 6322 Advanced Database Systems (Graduate level)

(`http://www.cs.cornell.edu/courses/CS6322/2008fa/`)

"This course covers recent research on the design and implementation of scalable data-centric systems. Topics include infrastructure for cloud computing, novel database architectures such as column stores and main-memory data management, the convergence of search over unstructured data and querying of structured data, power-aware data management, and data management for computer games and virtual worlds."

Duke University

CPS 216: Advanced Database Systems

(`http://www.cs.duke.edu/courses/fall07/cps216/`)

“Effective collection, analysis, and maintenance of data is key to achieve rapid progress in almost all disciplines of science and engineering. In this course we will cover the core principles and techniques of data and information management. The potential topics covered in class include processing and optimization of declarative queries, transactions, crash recovery, self-tuning database systems, data stream systems, information retrieval and Web data management (e.g., Internet search engines like Google), and data mining. The course materials will be drawn from textbooks as well as recent research literature.”

ETH Zurich

252-3002-00L: Algorithms for Database Systems
(http://www.vvz.ethz.ch/Vorlesungsverzeichnis/lerneinheitPre.do?lerneinheitId=75594&semkez=2012S&lang=en)

“Objective: Develop an understanding of selected problems of current interest in the area of algorithms for database systems. Topics: Query processing, optimization, stream-based systems, distributed and parallel databases, non-standard databases.”

University of Edinburgh

Advanced Databases (Graduate Level)
(http://www.inf.ed.ac.uk/teaching/courses/adbs/)

“Overview, introduction to query evaluation. Introduction to relational databases, indexing, B+trees, extendible hashing, Linear hashing, R-trees. External mergesort, Physical plans. Execution models, nested loops join Graefe's paper on query evaluation in relational databases. Nested loops join (cont.), sort-merge join, Hash join, introduction to query optimization. SQL decomposition, equivalence rules. Original System-R paper. Histograms. Dynamic programming, further optimization algorithms. Transaction processing. General locking algorithms, B+-tree locking. Transaction isolation, introduction to recovery, ARIES recovery algorithm, Parallel database systems”.

7 Advanced Database Research

In the following sections we list the research groups which have had historically a major impact in the database field.

7.1 Academic Research Groups

University of Wisconsin-Madison

Database Systems Group

`https://database.cs.wisc.edu/`

Stanford University

InfoLab

`http://infolab.stanford.edu/`

MIT

Computer Science and Artificial Intelligence Lab

Database Group

`http://db.csail.mit.edu/`

University of California, Berkeley

Berkeley Database Group

`http://db.cs.berkeley.edu/w/`

Carnegie Mellon University

Database Group

`http://www.db.cs.cmu.edu/db-site/`

Cornell University

Database Group

`http://www.cs.cornell.edu/bigreddata/`

CWI Amsterdam

Information Systems Group/Database Architectures

`http://www.cwi.nl/research-groups/Database-Architectures`

7.2 Industry Research Groups

IBM T. J. Watson Research Center

Database Research

`http://www.research.ibm.com/scalabledb/`

Microsoft Research

Database Group

`http://research.microsoft.com/en-us/groups/db/`

8 Bibliography

This section lists references in different categories. First, the textbooks. For most of the topics in modules 2 and 3 the Ramakrishnan and Gehrke (2003) book is proposed, but students can also use the Silberschatz, Korth *et al* (2010) book. The Bernstein, Hadzilacos, and Goodman (1987) book is proposed because it is downloadable in pdf format and has a comprehensive and standard treatment of serializability. The Date (2003), Elmasri and Navathe (2010), Silberschatz, Korth *et al* (2010), and Garcia-Molina, Ullman *et al* (2008) are available at the UFP libraries. The Chaudhri and Loomis (1998) is also available at the libraries. As for the research papers most are also available from the digital library, and a copy is provided in the course platform.

The second set of references lists efforts to define a body of knowledge and curricula in several computing areas, notably the ACM/IEEE initiatives. We listed also some papers discussing the database curriculum, present and future.

The third set of references lists several papers, dealing also with database education in general, that were deemed interesting. Some papers analyze using code from real-world systems (e.g. PostgreSQL) to teach database internals. Others discuss how we can use automated tools to do automatic grading and give constructive feedback to students. Grading SQL code is a hard task, and the authors present several tools. Some papers discuss using large, real-world data sets and commercial services like Amazon web services. Other papers discuss group work and student satisfaction, and teaching approaches. Finally, we include a set of references discussing security and databases.

8.1 Text books

Bernstein, P. A., V. Hadzilacos, N. Goodman (1987). **Concurrency Control and Recovery in Database Systems**, Addison-Wesley. Online at http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx.

Blaha, M., Premerlani, W. (1998). **Object-Oriented Modeling and Design for Database Applications**, Prentice-Hall.

Date, C. (2003). **An Introduction to Database Systems**, 8th ed., Addison Wesley.

Elmasri, R. and S. B. Navathe (2010). **Fundamentals of Database Systems**, 6th ed., Addison-Wesley.

Garcia-Molina, H., J. Ullman *et al* (2008). **Database Systems: The Complete Book**, Prentice Hall.

Gray, P., K.G. Kulkarni, and N.W. Paton (1992). **Object-Oriented Databases: A Semantic Data Model Approach**. Prentice-Hall, Hertfordshire.

Hellerstein, J. and M. Stonebraker (2005). **Readings in Database Systems**, MIT Press.

Chaudhri, A., Mary Loomis (1998), **Object Databases in Practice**, Prentice-Hall.

Ramakrishnan R. and J. Gehrke (2003). **Database Management Systems**, 3rd ed., McGraw-Hill.

Silberschatz, A., H. Korth *et al* (2010). **Database Systems Concepts**, 6th ed., McGraw-Hill.

8.2 Database body of knowledge and curriculum

ACM, AIS, IEEE (2005). *Computing Curricula 2005. The Overview Report*, The Joint Task Force for Computing Curricula 2005, ACM, AIS, IEEE.

ACM, IEEE CE (2004). *Computer Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*, ACM, IEEE.

ACM, IEEE CSC (2008). *Computer Science Curriculum 2008: An Interim Revision of CS 2001*, Report from the Interim Review Task Force, ACM, IEEE.

ANECA (2005). *Título de Grado en Ingeniería Informática*. Livro Branco da Agencia Nacional de Evaluación de la Calidad y Acreditación, Spain.

Calero, C., Piattini, M., Ruiz, F. (2003). *Towards a database body of knowledge: a study from Spain*, **SIGMOD Record, 32**(2).

DBTEchNet Portal, http://dbtech.uom.gr/, visited 10 April 2012.

Martínez-González, M., Duffing, G. (2007). *Teaching databases in compliance with the European dimension of higher education: Best practices for better competences*, **Education and Information Technologies, 12**(4):211–228, Springer.

Murray, M., Guimarães, M. (2008). *Expanding the database curriculum*, **Journal of Computing Sciences in Colleges 23**(3).

Robbert, Mary Ann, Ricardo, C. (2003). "Trends in the Evolution of the Database Curriculum", *ITiCSE'03, Innovation and Technology in Computer Science Education*, 30 June – 2 July, Thessaloniki, Greece.

Springsteel, F., Robbert, M., Ricardo, C. (2000). "The Next Decade of the Database Course: Three Decades Speak to the Next", *SIGCSE'00 ACM Technical Symposium on Computer Science Education*, ACM Special Interest Group on Computer Science Education, 8-12 March, Austin, Texas.

8.3 Other references

1. The following papers present experiences and results of courses based on hands-on approaches to database teaching:

Ailamaki, A., Hellerstein, J. (2003). *Exposing Undergraduate Students to Database System Internals*, **ACM SIGMOD Record 32**(3).

Sciore, E. (2007). "SimpleDB: A Simple Java-Based Multiuser System for Teaching Database Internals", *SIGCSE'07, ACM Technical Symposium on Computer Science Education*, ACM Special Interest Group on Computer Science Education, March 7–10, Covington, Kentucky.

2. The following papers present tools to help students get feedback on their progress in some topics, like writing queries:

Allenstein, B., Yost, A., Wagner, P., Morrison, J. (2008). "A Query Simulation System to Illustrate Database Query Execution", *SIGCSE '08 Technical Symposium on Computer Science Education*, ACM Special Interest Group on Computer Science Education, 12–15 March, Portland, Oregon.

Barker, S., Douglas, P. (2004). "A Database Transaction Scheduling Tool in Prolog", *First International Workshop on Teaching Logic Programming: TeachLP 2004*, Saint Malo, September 8-9, 2004

Dekeyser, S., de Raadt, M., Lee, T. Y. (2007). "Computer Assisted Assessment of SQL Query Skills", *ADC2007, 18th Australasian Database Conference*, Ballarat, Australia.

Ullman, J. (2003). "Improving the Efficiency of Database-System Teaching", *SIGMOD 2003*, June 9–12, San Diego, CA.

Ramakrishna, M. (2000). "A Learning by Doing Model for Teaching Advanced Databases", *ACE 2000 Australasian Computing Education Conference*, 4-6 December, Melbourne.

3. The following papers discuss using large data sets:

Gudivada, V. N, Nandigam, J., Tao, Y. (2007). "Enhancing Student Learning In Database Courses with Large Data Sets", *37th ASEE/IEEE Frontiers in Education Conference*, 10-13 October, Milwaukee.

Wagner, P., Shoop, E., Carlis, J. (2003). "Using Scientific Data to Teach a Database Systems Course", *SIGCSE'03 ACM Technical Symposium on Computer Science Education*, ACM Special Interest Group on Computer Science Education, 19-23 February, Reno, Nevada.

Holden, E., Kang, J. W., Bills, D., Ilyassov, M. (2009). "Databases in the Cloud: a Work in Progress", *SIGITE'09 ACM Special Interest Group for Information Technology Education Conference*, 22–24 October, Fairfax, Virginia.

4. The following papers discuss several aspects related to teaching:

Drury, H., Kay, J., Losberg, W. (2003). "Student satisfaction with groupwork in undergraduate computer science: do things get better?", *5th Australasian Computing Education Conference (ACE2003)*, 4-7 February, Adelaide, Australia.

Iqbal, R., James, A. (2008). "Scenario-based Assessment for Database Course", *8th IEEE International Conference on Advanced Learning Technologies*, 1-5 July, Santander, Cantábria.

Oussena, S., Dunckley, L. (2007). "Adopting Student-Centred Approach to Advanced Database Teaching", *24th British National Conference on Databases*, 3-5 July, Glasgow University, Lecture Notes in Computer Science (LNCS) series. Springer Verlag.

Slazinski, E., D. (2003). "Teaching Data Warehousing to Undergraduates — Tales from the Warehouse Floor", *CITC4'03, 4th conference on Information technology curriculum*, 16–18 October, Lafayette, Indiana.

5. The following papers discuss database and security topics:

George, B., Valeva, A. (2006). "A Database Security Course on a Shoestring", *SIGCSE'06 ACM Technical Symposium on Computer Science Education*, ACM Special Interest Group on Computer Science Education, March 1–5, Houston, Texas.

Guimarães, M., Mattord, H., Austin, R. (2004). "Incorporating Security Components into Database Courses", *Information Security Curriculum Development (InfoSecCD) Conference'04*, 8 October, Kennesaw, Georgia.

Guimarães, M., Murray, M., Austin, R. (2007). "Incorporating Database Security Courseware into a Database Security Class", *Information Security Curriculum Development Conference'07*, 28-29 September, Kennesaw, Georgia.

Said, H., Guimarães, M., Maamar, Z., Jololian, L. (2009). "Database and Database Application Security", *ITiCSE'09, Innovation and Technology in Computer Science Education*, 6–9 July, Paris, France.

Srinivasan, S., Kumar, A. (2005). "Database Security Curriculum in InfoSec Program", *Information Security Curriculum Development (InfoSecCD) Conference '05*, 23-24 September, Kennesaw, Georgia.

Yang, L. (2009). "Teaching Database Security and Auditing", *SIGCSE'09, ACM Technical Symposium on Computer Science Education*, March 4–7, Chattanooga, TN.

9 Assessment examples

We list a sample of typical quizzes:

Schedulers:

1. are the schedules serializable? why? Use all methods available to answer.
    a. l1(x),l2(x),l3(y),e2(x),l1(y),e1(y),l3(x),e3(z)
    b. l1(x), e1(x), l2(x), e2(x), l1(y), e1(y), l2(y), e2(y)
    c. l1(x), l2(x), e1(x), e2(x), l1(y), l2(z), e1(y), e2(z)
    d. l1(x), e1(x), l2(x), e2(x), l1(y), l2(z), e1(y), e2(z)
2. test each one under a strict 2PL, TO and Optimistic scheduler.
3. which ones are recoverable, strict, and avoid cascading aborts?
4. could this Schedule be produced by TO:
    a. l4(y) l5(z) l1(x) l3(y) e3(z) e5(z) e3(y) l2(z) e4(x) l1(y) e1(x) l2(y) e2(z)
5. Use TO and check if these schedules are legal:
    a. l1(x); l2(y);l2(z); l3(y); c2; e3(y);e3(z)
    b. l1(x), l2(y);e2(x); c2; e1(y)
6. which anomaly occurs here: l1(x), e1(x), l2(x), e2(x), a1 . How to avoid it?
7. find two equivalente serial schedules to: e1(x), e1(y), c1, l2(x), l3(y), e2(x), c2, e3(y), c3
8. Is l3(y) e1(x) e3(y) l1(y) l2(x) e3(x) e2(x) conflict serializable? And view serializable?
9. Test in PostgreSQL, under diferente isolation levels and with 2 clients, the following schedules:
    a. T1, T2, e1(x), e2(x)
10. Repeate provoking a deadlock
11. Repeat but use increments (set x = x +1). What is the result?
12. Use Read Committed, 2 clients, and run:
    a. e1(x), l2(x), c1, l2(x), c2
    b. l1(x), e2(x + 15), c2, x = x + 10, e1(x), c1
13. Repeat but use "select ... for update". What is the effect and why?

Recovery, ARIES:

1. After a crash the following log is recovered. Using ARIES, complete the lines (Pi are pages):

LSN	LOG
00	Begin checkpoint
10	End checkpoint
20	Update: T1 write P1
30	Update: T2 write P3
40	T1 commit
50	T1 end
60	Update: T3 write P8

70	Update: T3 write P6
80	T3 abort
90	Update: T2 write P4
100	CLR: T3 (undo LSN 70), undoNextLSN=60
110	Update: T2 write P5
XXX	CRASH, RESTART

Explain the differences if the checkpoint had not been done.

Buffer management

1- Describe the PostgreSQL buffer management. Compare with MySQL's for example.
2- Read the file src/storage/buffer/freelist.c and propose changes for a different algorithm (LRU, MRU, CLOCK).
3- Describe the implications of buffer management in performance.
4- Describe the relationship between the Buffer Manager and the Recovery Manager.

Storage organization

1- Describe the main storage models (NSM, DSM, PAX).
2- Analyze those models from a cache efficiency point of view.
3- Propose approaches to data structuring (tables) to take advantage of each approach.
4- Explain how they work with variable length fields.
5- Explain query optimizations to deal with their characteristics.
6- Explain how compression can improve their performance.

Object-oriented databases

1- Design the OO model of the Manufacturing problem. Compare the relational and the OO models.
5- Implement the model in db4o. Change the model (schema).
6- Discuss versioning capabilities from the user and client programmer point of view.
7- Answer the same set of queries with a Java client. Compare with SQL.
8- Optimize the model. Define clustering strategies.
9- Work with Native Queries interface. How does NQ work? Compare with SODA.
10- Study the concurrency control mechanism. What can go wrong?
11- Propose application-side concurrency control. How does that differ from the system approach?
12- Propose concurrency control mechanisms for long transactions with objects

Spatial databases

1- B-Trees are balanced index structures. How to make an R-Tree balanced?
2- How do quadtree-like structures compare to R-Trees for database indexing?
3- Analyze the Google Maps algorithm using quadtrees.

4- Analyze the R-Tree implementation using GiST.
5- Propose a quadtree implementation using GiST, using the original SP-GiST paper.
6- Design and implement a city hall management database and propose rectangle and point in polygon queries. Consider public equipment.
7- Revise the previous queries if you would use PostGIS. Note the differences. What about topological relations that you could use?

Column-oriented databases

1- Design a relational model for a RDF domain, for example, FOAF or other linked data domain.
2- Compare the relational model with a triplestore model
3- Compare the column approach with simulations using row stores.
4- Justify why either is adequate to the triples problem.
5- Identify strengths and weakness of the relational implementation.
6- Draw the same model in a key-value store.
7- Explain how retrieval is done in a column-store.